Journal of the Association for Information Systems

JAIS

Special Issue

Hierarchy, Laboratory and Collective: Unveiling Linux as Innovation, Machination and Constitution

Tony Cornford
London School of Economics and Political Science
t.cornford@lse.ac.uk

Maha Shaikh
London School of Economics and Political Science
m.i.shaikh@lse.ac.uk

Claudio Ciborra
London School of Economics and Political Science

Abstract

This paper considers the question of how the Linux open source collective structures and organizes itself as complexity and uncertainty increase. The study focuses on Version Control Software adoption in the Linux Kernel collective and the controversies it entails. The analysis draws on theory drawn from Science and Technology Studies and Actor Network Theory to consider the processes by which technology comes to play a role as an active agent within the collective. Through this approach the study helps to reveal how organizing occurs and how restructuring around technical means is negotiated based on constitutional as well as technical and performance criteria. What emerges from the analysis is the strong collective agency to which nonhuman actors contribute, and thus, their place at the core of open source activity.

Keywords: *open source, version control software, organizational design, information systems, Science and Technology Studies, Actor Network Theory*

* Michael Wade and Kevin Crowston were the accepting senior editors. This article was submitted on 15th October 2009 and went through two revisions.

Volume 11, Special Issue, pp.809-837, *December* 2010

Hierarchy, Laboratory and Collective: Unveiling Linux as Innovation, Machination and Constitution

1. Introduction

This paper considers some battles and debates over efforts to re-structure the organization and information systems used by Linux Kernel developers[1]. The paper focuses on the protracted discussions between 1995 and 2003 over the use of some form of version control software (VCS) to manage computer code elements and code updates. The development of code and its incremental improvement lies at the core of the open source software process, so the use of specific technologies to support this is an enduring topic of concern to those who work on the Linux Kernel. The 1995 to 2003 discussions about VCS are seen here as a manifestation of the perceived need to organize the code development effort and perhaps to incorporate within it some new technical means, but the debates soon turned to broader questions concerning the structure of the collective and questions of adherence and challenge to the variously understood core values of open source. These concerns can be seen as an exploration of ideas that go well beyond open source software development and address a fundamental conundrum of modern life: that is, how can people and machines (who need each other) get along, by what rules and through what formalities of debate and decision making?

This study relates to an episode that is now in the past. People who work on the Linux Kernel today happily use a version control system named GIT (www.git-scm.org), but once upon a time, the idea of using or not using such software was highly contentious. Why would this be and what can we learn from revisiting these debates?

To answer the first part of the question as to why VCS was contentious, we find that, at the time, the use or non use of such software mattered to many people and raised multiple concerns including the efficiency of the code production process, the transparency of decision making, continued fidelity to the ethos of open source, and the continuity and integrity of the Kernel code base (Cornford *et al*., 2005). Exploring these concerns, how they were expressed, and to a degree resolved, tells us something particular about the way that those involved in Kernel development at the time understood their collective efforts and abilities, and their own stake in the work. Beyond this, the study probes a more enduring question that has been only lightly touched on in the research literature of open source: What role do artifacts such as VCS, but also bug tracker software, licenses, mailing lists, and wikis, play in shaping the distinctive performance that is open source software development? Our guiding research question asks, “How influential are these material and symbolic actors in steering such efforts and how is this influence manifested?” Put in sociological language, we ask, “What is their agency?”, where agency is defined as the ability to make a difference (Jones and Rose 2004).

The broader premise on which this paper is based is that we can learn from studying the ways the Linux developers worked out a sustainable collaborative arrangement when under stress and confronting hard choices regarding intertwined issues of organization design, information systems architecture, and core values, and as the role and relationship between code, software tools, and people was being reassessed. These arrangements were diverse, and over time, different ways of combining people and technical tools were tried and assessed. Some succeeded, and some failed, and this work to accommodate people and powerful software had the character of an experimental programme undertaken within a laboratory. Using the language of Actor Network Theory (ANT), these attempted arrangements (experiments) are “machinations” – attempts to hold together various competing interests through a material device or arrangement – something that is at the same time a technical artifact and a political strategy.

Given the complexity and novelty of the open source world, and the intertwining of the issues we deal with in this paper, we build our analytic framework and theorization carefully. As a preliminary, we explore the information processing needs of open source, considering it as any other organization that grows and confronts a complex environment (Ciborra, 1993, (see Chapter 3); Galbraith, 1974;

[1] The original version of this paper was written in 2004 together with Professor Claudio Ciborra. Claudio died in February 2005. We have substantially changed and revised the work since, but we would like to acknowledge Claudio’s profound impact on our understanding of ANT, open source and organizing. See Cornford *et al* (2005) for more details of his early influence on this work.

Galbraith, 1977). This exploration helps us to identify in more conventional terms the design challenges that the collective faces in devising, adjusting, and revising its *structure* in the face of changes in people, tasks, technology, and environment. It also helps us match information processing needs (and, thus, technologies) to organizational structure. We then build our primary analysis on Science and Technology Studies (STS) (Callon, 1986a; Callon, 1999; Latour, 1991; Latour, 1996; Latour, 1999a; Latour, 2004; Law, 1992; Law, 1997; Law and Mol, 1995), and, more specifically, Actor Network Theory (ANT). We use ANT first to reveal the dynamics of the interaction of actors, tools, and circumstances as matters of concern arise within the collective and various alternative socio-technical solutions (machinations) to these are asserted, debated, tried, and assessed – as *re-structuring* occurs (Weick, 1979) – and, second, as a way to reframe the issue of concern as one of respect for, and challenges to, the open source Constitution in the form of the GPL.

In section 3 we present this theoretical development in greater detail. Section 4 contains the case study of the LINUX Kernel development collective and describes the methodology used in collecting and analyzing the empirical data. Section 5 provides an interpretation of these events through the language and concepts of ANT. Section 6 reflects on the findings. The final section offers some concluding remarks on the utility of our new characterization of the collective and outlines some new research directions. But first, we introduce briefly the context of open source software development, the Linux project, and version control software.

2. Open Source Development and Version Control Software

Open source code development is defined in large part by its transparent process of collaborative development and the intellectual property regime and license that underpins it (Bruns, 2008; Chesney, 2006; Moglen, 1999; Perens, 1999; Suber, 2006; Suber, 2007). Numerous studies in the past decade have considered aspects of this software phenomenon: the economics of open source (Forge, 2000; Kollock, 1999; Lerner and Tirole, 2002), motivations (Hars and Ou, 2002; Lakhani and Wolf, 2005), and legal aspects (Fitzgerald and Bassett, 2003; Lessig, 1999). A smaller but interesting set of studies considers directly the question of organizing open source and its various governance structures (Crowston and Howison, 2006; Ljungberg, 2000). Demil and Lecocq (2006), for example, address the question of "Bazaar governance" set against neo-institutionalist ideas of hierarchy versus market versus network as seen through the lens of Transaction Cost Theory.

In this paper we add to the understanding of organizing and governance of open source by drawing on a less exploited body of theory, Science and Technology Studies (STS). We are not the first to draw upon STS theory to explore the workings of open source collectives (De Paoli and D'Andrea, 2008; Lanzara and Morner, 2005; Tuomi, 2001). De Paoli and D'Andrea (2008), for example, use Actor Network Theory to show how good code contributions are made possible and teased out of contributors through the inscriptions in artifacts such as version control software. Actors (developers) are obliged to behave and interact with the artifacts in a very particular manner, which "tests" their patches for the needed rigour. Their use of the ANT idea of translation is simple and effective.

There are also a number of studies that use boundary objects, a concept related to STS and ANT literature. These have a particular focus on knowledge creation and exchange in open source collectives (Bach, 2009); the creation of boundary institutions to exchange practices and rules (O'Mahony, 2005); organizing and conflict resolution through the use of boundary organizations (O'Mahony and Bechky, 2008); and collaboration and conflict negotiation (Jensen and Scacchi, 2005). O'Mahony and Bechky (2008) describe an attempted collaboration between various parties in an open source community (with divergent views) and a company intervention. They show that the creation of a boundary institution that behaved as an intermediary between the various parties proved to be successful, not because it blurred boundaries, but because it reemphasized them. Similarly, De Paoli et al. (2008) look at the debate over licenses in two projects, Geographical Information System GRASS and the OpenSolaris operating system, and use the idea of boundary objects to show how discussion about which license to adopt led to a greater understanding of FLOSS, and, thus, changed the character of the communities. They frame their understanding of boundary objects with ANT and are thus able to reflect on the key role licenses take as actors in the debate.

Lanzara and Morner's (2005) work on organizing in open source development addresses a wider account of technology's role in coordination of collective agency – the way that the collective behaves. The key question they pose is, "What role do artifacts and tools – technological or other – play [...] are they perhaps the hidden rulers and managers of open source projects?" In this, they draw on a number of different theoretical strands including Luhmann (1984) and STS literature (Akrich, 1992; Joerges and Czarniawska, 1998; Latour, 1991) to explain knowledge inscription through an examination of three different artifacts found within the collective – source code, mailing lists, and the copy-left license. Their conclusion, rather than their premise, touches upon the unexplored nature of software actors and their agency. This is where we begin our study.

2.1 The Linux project

At the heart of Linux activity, then as now, is the mission to develop an operating system kernel, the core code that links hardware to the various software functionalities that an operating system provides for programs and users The project to develop the kernel had been led from the start in 1991 by Linus Torvalds, starting out as a student in Finland, moving to the United States, and steering the project through to its position as a major force in the industry and an icon for open source (Weber, 2004). Torvalds is often referred to as the Linux benevolent dictator, and is the final arbiter of all questions and concerns.[2]

By the end of the 1990s Linux was the primary success story for open source and its advocates.[3] The demonstrated ability to produce what was understood to be the hardest and most complex type of software – an operating system – through the open source process was taken as a powerful indicator of the potential for this approach to challenge conventional software production models. It was also at this time that strong commercial interest began to be expressed in open source, in general, and in Linux, particularly, with, for example, IBM making a strategic decision to support Linux in 1999. There was something distinctive, more innovative, more transparent, and higher quality about open source code, or so its proponents claimed (Raymond, 1999; Perens, 1999; Moody, 2001). Linux had, almost from the start, been licensed under the GNU General Public License – the GPL – which allows distribution and sale of versions of Linux, but demands that such copies be similarly licensed and include distribution of their source code. This history places Torvalds' endorsement of the use of proprietary VCS software to help manage Linux (see below) in a clearer context. If open source was going to revolutionize how software was developed and managed, then why was Torvalds using and endorsing BitKeeper – a commercially licensed software?

2.2 Version Control Software

The usual starting point for a discussion of VCS is not with questions about its agency. Rather, VCS is conventionally presented as a technical system or application through which multiple elements of a software product can be corralled, assembled, reviewed, managed, kept track of, and protected against loss (Clemm, 1989; Kilpi, 1997). As well as accepting and locating code elements (patches), VCS stores and generates metadata about such elements and changes made, including details of authorship, time and date of posting, and explanatory comments. VCS also allows selected export of code elements, and in this way can allow a specific build (e.g., version) of a software system to be extracted, compiled, and used. Contemporary VCS is designed to work in a distributed Internet environment and to support (to some degree) parallel development (two or more people working on the same sections of code), and then to enable a subsequent resolution and consolidation of such parallel changes.[4] The earliest forms of version control software can be traced at least as far back to the original UNIX tools of the 1970s, in

[2] The term *benevolent dictator* was coined by Raymond in his essay *The Cathedral and the Bazaar* (1999). Some find it an offensive or challenging epithet, but in the open source world the benefits of harnessing embodied technical charisma to strong leadership is acknowledged – an echo of the early Enlightenment idea of the *Enlightened Despot.*

[3] The books by Moody (2001) and Weber (2004) provide an excellent summary of the Linux story in the period studied here. See also the information and timeline available on the linux.org web site http://www.linux.org/info/

[4] For a brief discussion of fundamentals of version control see *Version Control with Subversion For Subversion 1.5 (Compiled from r3305)*, Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. Available on-line at http://svnbook.red-bean.com/

particular, the *diff* and *patch* programs, and then to Rochkind's *Source Code Control System* (SCCS) and Tichy's *Revision Control System* (RCS) (Bolinger and Bronson, 1995; Rochkind, 1975; Tichy, 1985). This evolutionary path has formed the basis for most versioning software in use today (Fogel, 1999), including the two principal systems discussed in this paper, CVS (Concurrent Versions System) and BitKeeper (BK). Table 1 summarizes the main concepts used in this paper when discussing this technology.

Table 1. Core Concepts within the Debates on Version Control Software

Linux Kernel	The code developed by the LINUX collective, which serves as the fundamental management layer for the LINUX operating system. The Linux Kernel architecture is monolithic but modular.
Linus Torvalds	Originator of the Linux Kernel project, in 1991, and leader since then. His central role as the final arbiter of disputes has earned him the title of "benevolent dictator."
General Public Licence (GPL)	The copyright license carried by the Linux Kernel granting reuse and reproduction rights to all on the condition that tools or software incorporating the code be source-distributed on the same counter-commercial terms. The GPL is seen here as the constitution of the collective.
Version Control Software (VCS)	Software to manage code and coding activity. Synonyms include source code management and configuration management.
Tree	Structure to store code in multiple modules, nodes representing individual code elements. Developers may hold their own "tree." An official tree represents the primary account of the collective's work. A tree can be held in simple file structures or within a VCS.
Write Access	A tree can be shared with "read only" or "read-write" permissions. At the heart of the controversy is whether to allow other people "write" access to Torvald's tree (or is it the collective's tree?).
Concurrent Versions System (CVS)	A venerable open source version control system commonly used by open source development projects. By the mid 2000s it was seen by many as superseded by new systems such as Subversion, Bazaar, and GIT.
BitKeeper (BK)	A closed source version control system, in its own terms a distributed configuration management system. While targeted at open source developers, BitKeeper attracted criticism for its non open source license – hence spoken of as "commercially crippled" and a "Darkstar" project.
GIT	An open source VCS developed by Linus Torvalds subsequent to the events reported in this paper, and now in use by the Linux Kernel collective and others http://git-scm.com/about. GIT is a fully distributed model, with every clone a full-fledged repository with complete history.
Patch	An addition or alteration to a piece of code, usually as a remedy to an existing bug or misfeature or to extend functionality. A patch may or may not work, and may or may not eventually be incorporated permanently into the program. Patches are the fundamental unit of accretion to a code base.
Metadata	Data about data. The accompanying information about a patch, its author, time of submission, comments, interactions with other patches, etc.
Gateway	A route for code and patches between one version control software and another. Part of the story of Bitkeeper is the establishment of a gateway through which code patches in BitKeeper could migrate to the ideologically purer world of CVS. It did not work, was not used, and was abandoned.

CVS has a long and distinguished history and has been widely used in both open source and proprietary development. Indeed, it has sometimes been taken to *be* VCS. CVS emerged from RCS and was converted into C code and licensed under the GPL by Berliner (1990). The open source ethos is tightly inscribed within CVS, not least because it was one of the first software tools to come under the General Public License. Thus, CVS is understood to be compatible with an open and transparent process of software development, achieved through its ability to allow access to older versions, to proposed patches, and to diverging branches of software, thereby, making the evolution of code transparent and allowing the "eyeballs" to focus on the bugs.[5] BK, in contrast, is a proprietary product and was engineered specifically, as described here, for use in open source development to address the particular needs of Linus Torvalds (Henson and Garzik, 2002; Moody, 2001; Weber, 2004 (p197)). There are increasing numbers of other open source VCS developed and in use by the OS community, including Arch and Subversion (Collins-Sussman *et al.*, 2002), as well as a number of proprietary tools such as IBM's RationalClearCase.[6] During the period of time considered in this paper, most VCS used a centralized repository, from which users would "check out" a code section, work on it, and then check it back in. To avoid clashes as parallel changes are made, some human intervention is often needed when altered code (a patch) is remerged with the repository. Since the early 2000s, a strong movement for peer-to-peer or decentralized version control systems (DVCS) has arisen, seen in systems such as GIT, Mercurial, and Bazaar.[7] Indeed, GIT was developed by Linus Torvalds subsequent to the events discussed here for use by the Linux collective, and is also in wide use in other open source projects.[8]

The role of VCS is usually described in terms of a means to help developers share, accumulate, and manage code elements, documentation, etc., and to build complete versions of software based on choices of modules (Koch and Schneider, 2002; Lopez-Fernandez *et al.*, 2004; von Hippel and von Krogh, 2003). As such, VCS it is not often problematized as a key stakeholder, as a nonhuman actor with agency, able to substantially shape the software development organization on its own account. This tool view, in good currency in the technical literature, denies such software its own agency seeing it as just doing what it is asked and on the basis of the functionality that is inscribed into it by its wise designers. "*Its function is to keep track of changes made to the source code by project developers. It also stores the project's source code along with programmers' written comments that explain their work in detail*" (von Hippel and von Krogh, 2003). Seen as a tool, a VCS is designed to support certain tasks and activities, and the question of how this functionality or capacity affects the wider organizational environment is usually left unexplored (but see De Paoli and D'Andrea (2008) introduced above).

But a VCS might have a more a substantial and significant role as an active organizer, strong and persuasive, and demanding of a place and a voice within the politics of the collective, making its own appeal to the governing Constitution. our study is concerned with exposing the fundamental aspect of control that resides in any VCS, implicit in its name. Control and transparency are closely interlinked in any constitution; here, control can serve transparency (e.g., retaining history, recording change sequences, making code public), and transparency may serve control (messages are clearly passed, things done (or not done) are seen, lessons are learned, feedback can be applied), but the *transparency of control*, its fundamental constitutional aspect, is a more subtle matter. If VCS controls – and embodies the Constitution – then we must ask, "On whose behalf, in what ways, and to what ends?" For example, a VCS with an "efficient" (but who says so?) auto-merging algorithm to pull together disparate parallel changes (patches), may be a very powerful and persuasive actor. Indeed, it even has Linus Torvalds, the leader of the Linux Kernel collective, as a spokesman talking glowingly of its agency and how it (in this case the software BK) had arrived at the heart of the collective: "*It was actually a case of 2,300 times, different people had done updates in parallel, and they had to be merged. And I think about fifty of those were manual. Everything else was automatic*" (Torvalds, 2003).

[5] Linus' Law: "given enough eyeballs all bugs are shallow"
[6] http://www-306.ibm.com/software/awdtools/clearcase/
[7] http://arstechnica.com/open-source/news/2009/01/dvcs-adoption-is-soaring-among-open-source-projects.ars
[8] http://git-scm.com/; Torvalds discuses GIT, BitKeeper and CVS in May 2007: http://www.youtube.com/watch?v=4XpnKHJAok8

3. What is Going on? Theoretical Points of Departure

In our analysis, we frame open source activity in and around the Linux Kernel, as undertaken by a collective, more specifically a socio-technical collective that includes people but also a number of artifacts and devices both material (a VCS) and symbolic (an ideology). We choose to use the concept of a collective in contrast to the more usual "open source community," or "open source project" because we want to consider explicitly the combinations of people, things, and ideas, and we want to understand how this combination (collection/collective) works to achieve its goals. Drawing on Actor Network Theory (ANT), we see the collective as a socio-technical phenomenon, framed analytically as a heterogeneous network of negotiated and enforced relationships between its various elements (Law, 1992). This heterogeneity is embodied in ANT in a principal of general symmetry, which enjoins us to treat humans and nonhumans symmetrically as actors within such a network, privileging neither one nor the other (Latour, 1987; Latour, 1988). Thus, in the ANT tradition, we eschew any prior distinction between what is collected or included (or otherwise), be it social (human), technical or conceptual (Callon 1987; Latour 1987; Law 2000). In particular, we consider version control software as a type of software actor that seeks to join and participate in the collective.

Latour (1999b, p304) presents the idea of a collective simply as "the association of humans and non humans." Callon and Law (1995, p. 485) drawing on a performative ontology, propose a collective as "an emergent effect created by interaction of the heterogeneous parts that make it up… So it's the relations – and their heterogeneity – that are important. Relations which perform." Agency is, thus, understood as being collective and as emerging out of the heterogeneous mix. The word collective also recognizes the core capacity to collect and hold various elements together – in this case people, code, various software actors, bugs, patches, ideas – taking the "collect" quite literally. Indeed, the software actor we study – version control software – has an explicit collecting role to track, accept and retain code, patches and their associated metadata, and in doing so, to draw other elements together, for example, developers and bugs.

Using the concept of the collective to frame open source activity lets us see beyond what would usually be understood as the "organization" of any given open source project – for example, the degrees of hierarchy and specialization found, governance structures, the tools in use, functional divisions, or established work processes. Rather, we look to the processual and performative activities of collecting and including that lie at the core. In taking this approach, we draw on the tradition of organizational studies that throws into question the idea that organizations as nouns (or even as collectives) can be considered as "ontologically stable" objects (Clegg *et al.*, 2005). Rather it is the acts of organizing upon which we should focus.

As described above, in the period studied, Linux was emerging as a highly successful open source collective, perhaps the most successful, but this rapid emergence and high profile meant that it had to address highly complex, novel, and uncertain tasks in an unfamiliar non-firm and non-market context. To do this the members of the collective needed to find novel and stable new accommodations between the human and technical actors, and this required some experimentation. Thus, the collective worked, to a degree, as a laboratory, a place where experiments were undertaken and new ideas that mix together people, technical apparatus, code patches, ideologies, or social structures were tried out. A laboratory is a place where we create a microcosm of the world and where "scientists" can work to prove or disprove facts (Latour, 1983). Central to this activity is the attempt to stabilize an idea or concept – that is to produce a fact – as an accommodation of various interests, and to do this in a way that it can be returned to the world reinforced and made more powerful. The kind of stabilized "fact" that the laboratory tries to produce can be in different forms but often presents itself as a socio-technical device or arrangement that holds together the various interests. As Latour (1987 pp. 128-129) proposes it, *"The simplest means of transforming [a] juxtaposed set of allies into a whole that acts as one is to tie the assembled forces to one another, that is to build a machine. A machine as its name implies, is first of all, a machination, a stratagem, a kind of cunning, where borrowed forces keep one another in check so that none can fly apart from that group.*" Can this be a valid description of version control software?

To help maintain coherence – to remain collected and to accept a powerful machination that might help – the members of the collective (human and nonhuman) needed to draw on a common set of understandings about how things are to be done, what is allowed, and what is not – what we call the Constitution. We draw this concept of a constitution directly from Weber's (2004, p. 179, 2005) work on the Linux community and open source. Specifically, he proposes that the open source license (in the Linux case the General Public Licence or GPL) serves as a *de facto* constitution of the Linux collective: "... the core statement of the social structure that defines the community of open source developers who participate in the project" (Weber, 2004). In a similar way, Demil and Lecocq (2006) propose the copy-left license as fundamental to understanding the governance of open source. They quote Bonaccorsi and Rossi (2003): "Licenses are the most important institution in the governance structure of open source projects" (see also O'Mahony, 2003). Weber (2005) suggests that the core constitutional value that the GPL license proposes is that developers will be treated fairly if they join the collective (community), suggesting that fairness is basic to the open source movement's social contract that offers a mix of freedom, non-discrimination and pragmatism. Similarly, Lanzara and Morner (2005) speak of the "rules of reciprocity" that the license inscribes. Indeed, being treated fairly can be understood as central to participation in open source in the sense that participants can believe that their work will not be expropriated and exploited by others (Benkler, 2004; Henkel, 2004). In this paper, we go a little further and ask that the other, nonhuman, participants in the collective be treated fairly under the Constitution, including code, patches, and software actors. As we see below, if this core value appears to be violated, then some members of the collective, and their spokespersons, become very concerned.

3.1 Organization and Technology

As questions of control, authority, integrity, and even survival emerge within the Linux collective (see section 4 below), a fundamental set of disjunctions start to emerge. On the one hand, we see an "organization," not just a chaotic grouping or flow, facing classic problems as it grows and as its task set expands and seeks to absorb increased variety. But, then again, this is not an organization as we know it, or at least such a view is limiting in fundamental ways. For example, the motivations for participation by people and code, their expectations and guiding norms (ideology), and the rules and accommodations that hold them together are all in question.

The challenge the collective faces can be posed as one of matching technology to organization and organization to technology. The Internet allowed the Linux collective to emerge and may have deeply affected the nature of any demands for formal organization and structure, but it does not erase them. As Weber (2006, p116) notes, for Linux, success created dilemmas of organization and tested the fundamental viability of the open source model. Hence, the search for a structure to stand upon, a stable, shared, and understood pattern of relationships among the various members of the collective able to to support an orderly and productive division of labor. This structure needed to embody elements such as authority to make key decisions and opportunity to contribute to them; communication of such decisions; control and supervision of activities and/or outcomes and, above all, coordination among, especially in the case of open source work, highly autonomous and heterogeneous members.

As described in Raymond's colorful language, open source is no cathedral – in organization design language, not a multi-layered hierarchy or bureaucracy (Raymond, 1999). However, it is not just a chaotic marketplace as the term "bazaar" might evoke (Demil and Lecocq, 2006); nor a large-scale, integrated peer group or clan (Ouchi, 1980). It is conceptually, if we take the case of the Linux collective in the period of study, a *simple hierarchy*, with just two levels, a benevolent dictator, Linus Torvalds, sitting on the upper level, making all the key decisions about what goes into the Kernel and addressing disagreements that are brought to him, and below everybody else. Without some information processing innovations, this particular arrangement can perform adequately only for fairly routine tasks (low uncertainty) and with a limited number of members.

In the case of Linux Kernel development, thanks to sophisticated infrastructures permeating and

penetrating every phase of communication, coordination, control, and documentation, aside from the one of actual code production, and through principles adopted by Torvalds very early on such as "source code modularization," the collective can extend its boundaries, operate through fairly loose arrangements, and still keep a flat hierarchical structure in ways unheard of in non-electronic organizations (Applegate *et al*., 1988). This is one of the marvels of open source and the challenge it poses as an organizational model to the more traditional response of the business world to increasingly complexity: more layers and more bureaucracy, all served by hierarchy-reinforcing information systems, (think of yesterday's MIS and today's ERP) (Ciborra, 2000; Davis and Olson, 1985; Pollock and Cornford, 2004).

A pure information processing view identifies three efficient solutions available to the collective, the benevolent dictator, and his closest collaborators, as the scale of operation becomes larger, more complex, varied, and dynamic: 1) increase the number of supervisory levels; 2) delegate and manage by objectives; 3) keep the hierarchy simple, as it currently is, and support it with a "vertical information system" capable of increasing the bandwidth, so to speak, of the existing channels of communication, control, and coordination (Galbraith, 1974; Applegate *et al*., 1988). Option 1 makes no sense either ideologically or organizationally. And there are few margins for further delegation and decentralization, limiting the utility of option 2. Delegation has indeed happened with Torvalds *de facto* delegating to two lieutenants separate areas of expertise and responsibility when it was apparent in 1998 that he could not cope with the number of incoming patches and the type of patches, i.e., those related to the network functionalities (Weber, 2004, p. 116). The modular architecture of the Kernel and the definition of interfaces are also embedded manifestations (in ANT language, inscriptions) of the second option, with these design artifacts serving as "statements of objectives" and, thus, as a means to support controlled delegation.

The first two options have been used and are perhaps at their limits. What of the third? What might be sought is a smart accomplice that serves the needs of the top of the hierarchy for coordination and control, and allows the upper level to operate in a manner similar to that which has been so effective in the past and, moreover, has gained a lot of legitimacy (respecting and expressing the Constitution). Some form of version control software might be such an accomplice, and may be able to persuade others that it can serve not only as a key component in the software development toolkit, but also as a technology of coordination.

Thus, what the collective actually pursues, as described in section 4, matches what the model prescribes. Should we be surprised that the clever and innovative Linux practitioners know how to redesign their collective?

But any quasi-engineering model of organizational design that focuses purely on cognitive and informational aspects, important as they are, has limitations. We emphasize two. First, such models are mute about the implementation process, the choices, maneuvers, and devices, what we call machinations, needed to move from an identified ideal solution to any sustained innovation in the workflow. In this respect, restructuring the collective through the reforms identified above needs to be studied by invoking different, complementary, theories, that keep at the center of attention the role of technology. A straightforward organizational analysis of power, or models of the learning organization or the network organization do not serve in such an environment, for we need to stay very close to technology, even as we talk about the politics of implementation (Keen, 1981; Markus, 1983; Markus *et al.*, 2000; Markus and Robey, 1988). What we seek is an analytical framework that gives equal or indistinguishable importance to human agents and to the technology, but at the same does not neglect all the aspects of power, structure, legitimation, etc. (Hanseth and Monteiro, 1997; Orlikowski and Iacono, 2001). For this, we turn to research methods developed in the field of Science and Technology Studies.

The second limitation of the quasi-engineering models is perhaps more subtle, but of no less significance. Most models of organizational governance are conceived having in mind the business firm as a socio-technical system executing increasingly complex tasks in turbulent environments. But a business is an organization defined by precise "institutional coordinates," such as a bundle of

property rights (who owns the means of production, the results of labor, etc.) and the employment contract (that makes the authority relationship legitimate and defines who is a member of the organization and who is not and settles conflicts by *fiat*, etc.) (Demil and Lecocq, 2006). Here, instead, we deal with a minimal hierarchical organization defined by a radically different institutional context: membership is voluntary, no stable streams of revenue are directed anywhere, adherence to ideals and ethical values motivate, conflicts cannot be resolved other than by consensus, and members (people, code) are free to "fork out" (exit) at any time. The simple hierarchy of the collective is embedded in an assemblage of code and coders adhering to the Constitution that expresses distinct but untested values such as trust in a technical rationality ("let the code decide"), and, above all, the fairness implied by the General Public License (GPL) (Weber, 2004). A redesign of the hierarchy might be proposed as essential to guarantee efficient delivery of quality software, but how will the hierarchy, at the same time, represent and adhere to the Constitution – the institutional glue that gives the collective vitality and identity?

3.2 STS and Actor Network Theory

Returning to the disjunction introduced above, we are now able to frame it as, on the one hand, the firm-like arrangement in the hierarchy, and on the other, the polity of open source under the Constitution. No pure political science or sociology of power can account for this novel institution, a political body that operates through and is embedded in a sophisticated technical infrastructure. Indeed, these technologies are active members of this body, and the Constitution talks about what all its members (human and technical) should do and how they should behave. The organizational engineering view provides slim understanding of the strength or innovativeness of the Constitution, or even its fundamental role. Thus, we turn to Science and Technology Studies (STS) which has for three decades pursued an interest in following and theorizing the mixed up human and technical "imbroglios" that are such a fundamental feature of our world (Latour, 1993 p. 3; Latour, 2005; Law, 1991).

The Linux collective then reveals itself as a mesh of performative relationships (Law, 1999); the actors are myriads – Torvalds the benevolent dictator, CVS, patches, bugs, hackers, eyeballs, Kernels, evil corporations, naive sentiments, technical rationalities. None, taken in isolation, exist. For in ANT, to be an actor is to have or claim or work toward a network (Callon and Law, 1995), and any actor can, and usually does, make claim to more than one network and renegotiates these claims as they adhere to others. Networks can, of course, be strong or weak, and some actors will work for strength and a central role, while others may work for weakness and dissolution. Networks are not stable, and the threat of a breakdown is often real – in the case of open source, one such breakdown is given a specific name, a fork.

Questions about how such networks are created and sustained lead to the "sociology of translation" (Callon, 1986b). This considers how an actor convinces or entices others to join *a* network, and that "my goals are your goals," Processes of translation are fundamental for a network to gain strength through allies, and the stronger a network becomes, the harder it is for others to impede its progress. Callon (1986b), through the example of scientists, fishermen, and scallops in St. Brieuc Bay, explains four "moments" of translation. These are taken as distinct forces, but not inherently as a sequence, by which an actor persuades a number of other actors and through them their networks, to believe that membership (alignment) is important and useful to them. In Callon's case, the main actors are fishermen, shellfish, and scientists, and in ours, Torvalds, patches, VCS software, and hackers. These moments of translation, named as problematization, interessment, enrollment, and mobilization, form the basis for our analytical reading of the case study presented in section 5 below.

4. Case Study - Version Control and the Linux Bazaar

Our study starts as three competing "version controllers" vie for a position at the heart of the collective. These are Linus Torvalds, the founder and benevolent dictator of the project and human version controller; Concurrent Versions System (CVS), a long established open source software

product; and BitKeeper (BK), at the time a new proprietary VCS eager to take on the role. The question of including version control within the collective was a long standing controversy, and networks are built around controversies. We trace this story from June 1995 to June 2003. This covers the transition from no VCS to partial use of CVS, and then to use of BK. We do not consider subsequent events and the move to the GIT version control system in 2005.

4.1 Methodology

The principal source of data reported is from the Linux Kernel Mailing List [LKML] archive, supported by other accounts of the period and other relevant online archives and discussion lists. The Linux Kernel Mailing List (LKML) archive is held at the University of Indiana www.uwsg.indiana.edu/hypermail/Linux/Kernel. Data was collected starting from posts made in the first week of June, 1995, through June 30, 2003. We conducted a systematic search through the use of key words, study of message threads and cross references, and following links and debates to identify discussion related to VCS. In the period from 1995 to 2003, there were 249 threads identified as discussing VCS, and 3,352 messages posted. An average thread consisted of 13 postings, and the most extended reached 320. Table 2 describes the 10 most extensive threads in our sample, shown in date order, and indicates the continuing, even growing, scope of the debate.

Table 2. The 10 Longest threads Relating to VCS in Date Order 1998 - 2003

Rank	Date	Title of Thread	Description	No. of Msgs
9	September 1998	2.1.123 and fbcon.c	What began as a small complaint about a patch soon flared into a serious argument in which Torvalds took active and aggressive part. This was around the time when there were rumors of him not being able to "scale," and being aware of this, he can be seen to demand more respect and control. However, he ends up insulting his trusted lieutenant, David Miller, and tops that with heavy abuse for VGER, the CVS server and tree updated by Miller. Torvalds makes a clear demand to Miller to shut VGER down as he claims it "keeps me out of the loop."	73
8	November 1998	The history of the Linux OS	A desire to preserve the historical evolution of the Linux operating system spurred a developer to ask the community to send him details about significant emails that were sent in the past, personal details about contributors and contributions, and even anecdotal summaries and photos. The amount of interest this request generated is a good indication of a strong desire to archive, and to access the archive almost as organizational memory. Another thread around this time was The Linux Kernel Compilation Project [proposal]. Interestingly, CVS was brought up as a way to preserve history. McVoy, who at that time was developing BK, made an attempt to advertise his software but most of the developers ignored and continued with CVS, and even RCS ideas.	87

5	December 2001	The direction Linux is taking	This is another thread concentrated on an urgent need for a VCS for the Kernel. The conversation steered into what good maintainership should be, but was later used by McVoy to promote BK. Oddly enough, it ended with some developer, who had obviously not been following this thread, asking why they don't use CVS!	118
1	January 2002	A modest proposal -- We need a patch penguin	A strong need for a VCS is voiced as Torvalds drops patches and developers become agitated that they and their code is ignored. Torvalds advises the collective not to "try to come up with a patch penguin." Instead, he advises, "try to help existing maintainers, or maybe help grow new ones. THAT is the way to scalability. He added that "development is done by humans," which makes a claim for human supremacy over technology. Is this a true picture of the reality, or not? The latter half of the messages are focused on BK, where we see McVoy make a strong attempt to enroll Torvalds into using his software for Kernel development.	320
2	April 2002	[PATCH] Remove Bitkeeper documentation from Linux tree	BitKeeper documentation is inserted into the Linux Documentation directory, and this caused an uproar among those opposed to BK, but also with some BK supporters. The Kernel developers want it made clear that the Kernel is not too intimately linked with a closed source software like BK. Proprietary software is looked upon with distaste and extreme mistrust. The worry is that BK will contaminate the open source code of the Kernel.	181
4	April 2002	BK, deltas, snapshots and fate of pre…	This is a continuation of [PATCH], Remove BitKeeper, where the discussion continues about ideology, proprietary software, and possible adulteration of Linux code. This is the time of early BK use and the developers are keen to put up a fight to rid the Kernel of it.	123
3	October 2002	New BK License Problem?	A clause is added to the BK license, which makes it illegal for any developer working on a BK competing software to use BK. This forces the developers to reassess questions about openness. The developers wonder if McVoy and his company intend to creep in more such subtle changes, and as trust is very slender between McVoy and the developers, it is not surprising that McVoy's every move is closely watched, speculated upon, and usually not given the benefit of the doubt.	144

10	October 2002	Bitkeeper outrage, old and new	Richard Stallman initiated this thread with his obvious call to the Linux Kernel collective to beware of BK and its proprietary license. BK supporters rallied around in defense of BK and McVoy. Their response was based on the claim that Stallman, by telling other developers to not use BK, was infringing on the collective's freedom to make its own choices.	70
7	February 2003	BitBucket: GPL-ed KitBeeper clone	This thread emerged from Machek's announcement of the BitBucket [BB] project, a self-professed clone of BK. BK users and supporters gathered together to crush this move through technical arguments of BK superiority. This leads into a discussion of the merits mostly of BK, but also of other VCS like SubVersion and Arch. BB never materialized but managed to cause a stir with a simple announcement.	106
6	March 2003	[ANNOUNCE] BK->CVS (real time mirror)	The focus of this thread was the announcement of the BK→CVS gateway by McVoy, however, this soon degenerated into another argument between CVS and BK users. The CVS users complained that metadata written by the developers was not complete in the gateway version of CVS. This thread linked into another, rather colorfully named as, Constant BitKeeper bitching.	97

The LKML archive breaks threads and messages down into weeks per year. On average each week had at least two threads of varying intensity on VCS issues. Our detailed reading of this collection of emails gave rise to three broad "eras." Each era marking a change in either the adoption of VCS, the use of it, or both. The rest of this section is structured around this historical reading.

The LKML can be seen as one of the laboratories in which issues related to development practices are extracted from the world and begin to be considered as matters of concern and then, through debate and collective sense-making, may be made matters of fact and associated with new or innovative machinations. As our story narrates, VCS was, indeed, such a matter of concern, but one that stubbornly for more than eight years refused to become a matter of fact or be associated with an enduring machination. It was only in mid 2005 that the software GIT was initiated and went on to gain the legitimacy that neither CVS nor BK discussed here could earn.

4.2 Pre-VCS to Partial VCS

In the mid 1990s, coordinating Linux as a small project was simple. If maintainers could keep up with patches, and not lose or confuse them, then VCS was not needed. Thus, Torvalds decided in 1995 to *not* adopt CVS as a core technology of the collective embedded in the process of Kernel development, and this decision was backed by his trusted lieutenants, Alan Cox and David Miller. "*I'm afraid that I don't like the idea of having developers do their own updates in my Kernel source tree. I know that's how others do it, and maybe I'm paranoid, but there really aren't that many people that I trust enough to give write permissions to the Kernel tree. Even people I have worked with for a long time I want to have the option of looking through their patches _first_, and maybe commenting on*

them (and I do reject patches from people)" (Torvalds and Schlenter, 1995, 1996).[9] As Moody describes, Miller did not remain happy with Torvalds disregarding the wider use of CVS (Moody, 2001) and, by late 1997, CVS was widely, if unofficially, used within the Linux collective as a means to share code and patches. Early messages in 1997 (April) proved to be the precursor to a heated debate over Torvalds' decision to not use CVS, and September 1997 saw David Miller's post announcing, "*This is a new thing I'm going to start doing", a decision to make 'full raw snapshots of my tree on vger available for ftp on a somewhat regular basis from now on. It contains also a copy of the full ChangeLog file from my CVS repository, it goes real far back, back to 1.99.x something"* (Miller, 1997 - Sun, 14th Sep)[10] (see also Weber 2005, p. 117). CVS then had an ally close to the top, and became more acceptable within parts of the collective. Torvalds himself was never convinced of CVS use, indeed was hostile to it,[11] so the tree that Miller kept up to date was only quasi-official.

4.3 Proprietary VCS, enter BK

The years 1998-1999 were probably the roughest years for the Linux Kernel collective because it was not able to cope with the workload. Torvalds was not using CVS or any other VCS. This led to unofficial CVS trees of the Kernel proliferating, causing confusion and uncertainty among the developers. The time was ripe for the emergence of a rival, a rival claiming to be *the* one true VCS. In February 1999 Larry McVoy announced, "*Most of you know we've been working on the next gen revision control for the Kernel for a while now. For those of you who don't know, BitKeeper is an Open Source distributed revision control system which I claim is a substantial step forward from CVS*" (McVoy, 1999 - Sun 21st Feb).[12] BK was the product of McVoy meeting with Torvalds and few of the other senior Kernel developers. CVS was not efficient enough for Torvalds, and there was a growing crisis in 1998 around Torvalds "dropping patches," with a long thread of messages focused on the topic of "Linus doesn't scale."[13] Indeed, there was talk about the Kernel forking. McVoy took this opportunity to explain how BK would resolve many of the issues Torvalds was facing. "*Linus, Dave Miller, and Richard Henderson came up to my house for dinner and we drew pictures on the floor for about 3 or 4 hours, and when we were done, Linus said "yeah, that's cool, if you build it and it works like you say, I'll use it". And I foolishly said "No problem, I've done it before, 6 months or so". That would have been around the fall of '98 I think*" (Andrews and McVoy, 2002).[14] McVoy was willing to create software that would closely match the specific needs of Torvalds because, as he said, "*Linus is what makes Linux great. He's the glue that holds it all together. Without him, Linux would splinter like the BSDs have*" (Andrews and McVoy, 2002).[15]

Torvalds was interested but didn't give a definite reply. He insisted that he needed to use the software and then make his judgement. When McVoy's company, BitMover, did finish the first version of BK in the late 1990s, it was not Torvalds who was the first user. "*For a couple of years before Linus ever touched it, Cort [Cort Dougan of FSM Labs and the Linux PowerPC Group] and his team used it. They jumped on it and used it when it was very early (in the development process) and were just incredibly helpful helping us debug problems. If there was some issue, they always saved the repository so we could go poking through it and try to figure out what was going on. Those guys you know they have done more for us than anybody else*" (Barr and McVoy, 2003).[16]

BK was not, however, universally seen as a remedy, and, for example, inspired the Darkstar project -- not being open source (GPLed) was anathema to many Linux developers. When an email from

[9] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/9602/0800.html and see http://www.youtube.com/watch?v=AxkGKtVNdik
[10] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/9709.1/0432.html
[11] see http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/9809.3/0766.html "I'm _really_ disappointed by how the thing has been handled, and very very disappointed by vger."
[12] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/9902.2/0812.html
[13] http://www.uwsg.iu.edu/hypermail/Linux/Kernel/0201.3/1000.html
[14] http://Kerneltrap.org/node.php?id=222
[15] http://Kerneltrap.org/node.php?id=222
[16] http://www.Linuxworld.com/story/32618_4.htm

Berglund[17] (Berglund, 2000 - Mon 11th Sep) characterizing BK [indirectly] as a Darkstar was sent on Monday, September 11, 2000 at 12:13:56 EST, it took McVoy less than a minute to respond [12:14:08 EST]. Torvalds, however, didn't think the license would pose such an issue, and in February 2002 declared BitKeeper the official version tool for the Kernel (Torvalds, 2002 - Tue 5th Feb).[18] A volley of dissenting voices were heard on LKML, including, "*Linux ceases to be free software when you require nonfree software to contribute it*" from a trusted lieutenant, Alan Cox (Cox, 2000 - Wed 13th Sep).[19]

The situation was exacerbated when, in early October 2002, some developers noticed that the BitKeeper license (BKL) had been changed to include a new clause: "...this License is not available to You if You…develop, produce, sell, and/or resell a product which contains substantially similar capabilities of the BitKeeper Software, or, in the reasonable opinion of BitMover, competes with the BitKeeper Software." Gall said, "*This would seem to be a change which is not Open Source developer friendly*" (Gall, 2002- Fri 4th Oct).[20] But McVoy replied, clarifying that BitMover (BM) had left out the word "distribute" in the clause in order to protect open source developers (McVoy, 2002 - Fri 4th Oct).[21] The substance of concern about BK not being GPLed is well expressed by Molnar: "*Today the 'Linux Kernel' is not the source code anymore, it's the source code plus the BK metadata… by default the data and the MetaData is owned by whoever created it. You, me, other Kernel developers. We GPL the code, but the metadata is not automatically GPL-ed*." (Molnar, 2002 - Sun 6th Oct).[22] Molnar stressed that a change in the BKL was needed that "*ensures that metadata attached to GPL-ed code is also licensed under the GPL, and creates a clearly GPL-ed repository*."

In spite of all the problems, by early 2002, BK was the official tool for Linux, thanks to the support of Torvalds and a few of his lieutenants. This rosy period was not to last, and later that year BK was labelled on the LKML "*evil* corporate software." Richard Stallman took the opportunity to invoke his ideology of free software, only to be told by Torvalds that he doesn't believe in ideology. "*Quite frankly, I don't _want_ people using Linux for ideological reasons. I think ideology sucks. This world would be a much better place if people had less ideology, and a whole lot more "I do this because it's FUN and because others might find it useful, not because I got religion"*" (Torvalds, 2002 - Sat 20th April).[23]

4.4 Dual VCS the BK→CVS gateway

BK began to gain ground, and a number of Kernel developers adopted it. This tentative peace broke down when, in February 2003 Pavel Machek announced that he had decided to start a project called BitBucket, "*[a] bitkeeper clone*" (Machek, 2003 - Wed 26th Feb).[24] Not surprisingly, Larry McVoy retaliated quickly, "*BitKeeper is a trademark, please don't use the BitKeeper name when describing BitBucket*" (McVoy, 2003 - Sat 1st March)[25], because saying "*'BitBucket is a GPL-ed clone of BitKeeper' … implies that BitBucket does what BitKeeper does and nothing could be farther from the truth*" (McVoy, 2003 - Sat Mar 01).[26] McVoy's reply to Machek then began a long string of emails, most poking fun at BK and McVoy's adherence to proprietary software. Serious issues grew out of this discussion; for example, that the developer base was getting diluted by initiating too many version control projects and, thus, not creating any truly useful one. Garzik called this the "SourceForge Syndrome" (Garzik, 2003 - Sat 1st March),[27] recalling how a real conversation is needed before initiating any new tool Garzik brought up McVoy having dinner with Torvalds and others as an example (Machek, 2003 - Mon 3rd March)[28] (Bradford, 2003 - Sun 2nd March)[29].

[17] http://www.uwsg.iu.edu/hypermail/Linux/Kernel/0009.1/0472.html
[18] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0202.0/0989.html
[19] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0009.1/1076.html
[20] http://www.uwsg.iu.edu/hypermail/Linux/Kernel/0210.0/1496.html
[21] http://www.uwsg.iu.edu/hypermail/Linux/Kernel/0210.0/1500.html
[22] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0210.0/1918.html
[23] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0204.2/1018.html
[24] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0302.3/0931.html
[25] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0052.html
[26] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0057.html
[27] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0147.html
[28] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0422.html
[29] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0411.html

Hellwig suggested that projects should be encouraged to keep their infrastructure free and open in fidelity to the open source constitution (Hellwig, 2003 - Sat 1st March).[30]

Torvalds made a contribution to this discussion only near its final stages, when he spoke strongly of the technical virtues of BK (Torvalds, 2003 - Tue 4th March).[31] "*You've got at least 20 actively developed concurrent trees with branches at different points. Trust me. CVS simple (sic) CANNOT do this. ... Give it up. BitKeeper is simply superior to CVS/SVN, and will stay that way indefinitely since most people don't seem to even understand _why_ it is superior*" (Torvalds, 2003 - Fri 7th March).[32] McVoy took this as encouragement to give a more detailed explanation of how BK is more technically efficient and useful to the Kernel developers. Perhaps he felt that he had made little headway, and less than a month later McVoy surprised developers by creating a gateway between BK and CVS. This was not exactly encouraging a competing VCS, but did allow users of CVS and other VCS more access into what was happening in Linux development. McVoy summed up his company's motivation, *"[The] goal is to have the freedom to evolve our file formats to be better, better performance and more features. SCCS [a file standard shared by CVS and BK] is holding us back… The payoff for you is that you have the data in a format that is not locked into some tool which could be taken away. The payoff for us is that we can evolve our tool as we see fit."*

This move was eyed with great scepticism. The slender trust between the open source developers and the proprietary BK, in spite of McVoy's move to create the gateway, almost broke down again. Ben Collins from Debian expressed the main concern: "*You are giving us approximately 90% of our data in exchange for the one thing that made using Bitkeeper not a total sellout; the fact that the revision history of the repo was still accessible without proprietary software. I honestly appreciate the work that you and BitMover do for the Kernel, but not giving us access to 100% of _our_ data is unacceptable to me*" (Collins, 2003 - Tue 11th March).[33] McVoy's reply showed exasperation: "*None of us at BitMover would shed a tear if you moved off BK. This has *not* been a pleasant experience for us*" (McVoy, 2003 - Wed 12th March)[34] because "*The main thing is that the CVS server and the tarball of the CVS repository are *not* under our control. That's the only way some people are going to believe that we're not out to screw them and it would [be] oh-so-nice to have people think that, it really would*" (McVoy, 2003 - Tue 11th March).[35] McVoy insisted that nothing was missing from the data made available through the gateway and that developers were just being paranoid and mistrustful, "*we actually captured 100% of the checkin information, both in data files and in the pseudo ChangeSet file, not one byte of that is lost. All we did is collapse all the branches into the longest possible straight line, which is actually for many purposes nicer than the rats nets that you get [within] BK*" (McVoy, 2003 - Wed Mar 12).[36]

McVoy's show of ownership of the BK metadata disturbed a number of developers, "*As for the data, you are right, we don't own that. As for the metadata which makes BK work, that's ours, not yours. BK made that metadata, you did not. If you don't like those terms, convince Linus and friends to get off of BK. That would be just fine with us*" (McVoy, 2003 - Sun 16th March).[37] A day later McVoy announced that the BK to CVS gateway had gone live and could now be accessed (McVoy, 2003 - Mon 17th March).[38] This is where we leave the story; however, not long after, the breakdown of this gateway was reported "*due to disuse and security problems*" (Anvin, 2004 - Thu 22nd July).[39] The final break between BK and Linux occurred in spring 2005 and is detailed at http://Kerneltrap.org/node/4966.

[30] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0092.html
[31] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/0695.html
[32] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0303.0/1932.html
[33] http://www.ussg.iu.edu/hypermail/Linux/Kernel/0303.1/0894.html
[34] http://www.ussg.iu.edu/hypermail/Linux/Kernel/0303.1/1242.html
[35] http://www.ussg.iu.edu/hypermail/Linux/Kernel/0303.1/0900.html
[36] http://www.ussg.iu.edu/hypermail/Linux/Kernel/0303.1/1134.html
[37] http://www.ussg.iu.edu/hypermail/Linux/Kernel/0303.2/0121.html
[38] http://www.ussg.iu.edu/hypermail/Linux/Kernel/0303.2/0219.html
[39] http://www.uwsg.indiana.edu/hypermail/Linux/Kernel/0407.2/0490.html

5. Interpretation and Analysis

This case is rich with detail concerning how version control software was understood to potentially play a focal role in the way an open source project is organized, its politics, and its productivity. There are many ways to approach the story. Our approach is to ask why Torvalds does not want CVS, and whether his mind can be changed.

5.1 Why Torvalds does not want VCS – problematization

Linus Torvalds has the heavy responsibility of reviewing every patch that is sent to the Linux Kernel for inclusion, so why doesn't he accept some help? He was over time so burdened that he delegated part of his work to trusted lieutenants. These individuals gained their position by showing great expertise in specific areas – gaining Torvalds' trust. However, more aid was needed because Torvalds was still receiving numerous patches. CVS is interested in becoming a part of the Linux Kernel development, as its various spokespersons make clear, and offers benefits to Torvalds, the code, and the Kernel developers.

Problematization, or "how to become indispensable," is the required process of relationship building to convince others that there is, or could be, a mutual goal; and in order for all to succeed, something needs to be done. An actor, in this case CVS, seeks to become indispensable to others, or in ANT terms, an obligatory passage point [OPP] through which all other actors must pass. What can CVS offer to the other actors?

Torvalds wants to be able to maintain tight control over what is included in the Linux Kernel. For a number of years, he has managed this process through email submissions of patches by developers and offering no write permissions to his tree. However, this was before the Kernel development grew too large to be controlled without some aid. Partial delegation has helped, but he has come to a point in his life where a new job, baby, and Linux are proving impossible to juggle – as Larry McVoy phrased it in a post in September 1998, "*Linus doesn't scale."* Torvalds stress is beginning to show, he is ignoring patches, and patches do not like to be ignored and tell their authors. Torvalds knows that this could cause a fork and ruin what he has worked so hard for. He has thought about using CVS (or another VCS), but his experience has convinced him that this is not the route to reestablish the Kernels trajectory or his control. Unlike the lieutenants, Torvalds doesn't trust CVS.

The Kernel needs to grow bigger, better, and more effective, to beat competitors and perpetuate itself by becoming the operating system of choice. It can only do so if the patches keep coming, and that requires that the collective is busy and buzzing. Torvalds' use of some form of VCS might be useful to the Kernel, but then again, it could be a prison in which the Kernel is locked up or exiled. VCS are unpredictable in effectiveness and functions and raise difficult issues about who should have write access. Thus, it is questionable whether this route will really speed up the processing of patches. If there is to be a VCS, can the developers and Torvalds decide which one they will ally with?

Kernel developers want to work on the Kernel -- in Torvalds' exhortation, they want "to have fun," -- or is their motivation to deploy their ideology, or peer recognition, or to get a better job? Some are already using CVS publicly, and the overall feeling among the majority of the developers, as their spokespersons claim, is that the Linux Kernel should be specific; there should be one VCS that everyone can access and use.

Patches sport their own habits: They want to be read, reviewed, amended, and above all accepted. How can they achieve this? By being paraded in front of Torvalds, by being put on public display (peer review), by being tested and improved. They also need to locate their bug, (remember, an actor makes a claim to a network) and some form of VCS might serve them if it could perform as a bug producer alongside a role as a patch parade. Patches may choose CVS as their preferred VCS, given that their authors (developers) are often experienced users, but in the end, they probably don't care. What the patches do not know at this point is that their status (and of their metadata) under the GPL may be brought into question if the chosen VCS is not adhering to the Constitution.

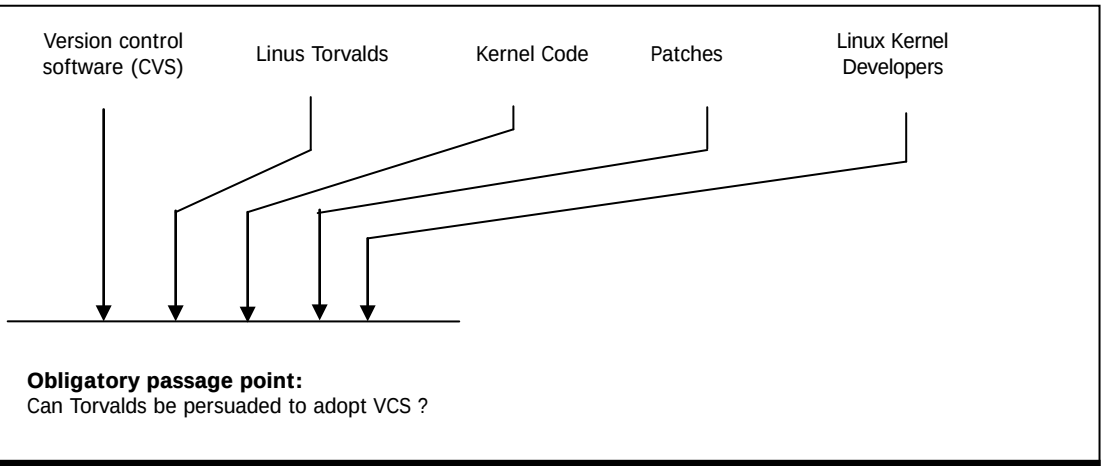

Figure 1. VCS Trying to Become an OPP

The various actors and their spokespersons must craft a response to all these potential allies. This all seems to give rise to one question that links all the actors: "Can Torvalds be persuaded to adopt VCS, and if so which flavor?" [see Fig 1]. CVS is in a position to entice some of the other actors to contribute to its own goals, to recognize it, and use it as the channel through which their ends might be met (e.g., Millar's *vger* tree); in other words (the words of ANT), to make the obligatory passage point not Torvalds but VCS. But it could not summon the strength, the allies, to make it happen. Here, the network rested for a number of years – Torvalds could not be persuaded! But then BK emerged, with its spokesman Larry McVoy.

5.2 Why Torvalds accepts VCS – interessement

Interessement, or "how the allies are locked into place," is about strengthening and enforcing commitment in order to guard against appropriation by others of useful actors and their networks. As Callon (1986) explains, interessement refers to being "in between" something. In other words, it relates to indecision: There is usually more than one influence operating at a time and more than one possible way to go. If Torvalds accepts the use of CVS, or any other VCS, it implies that the competing influences that would have persuaded him otherwise have been weakened. Torvalds requires the assurance that he will always have ultimate control over the Kernel -- indeed, he resisted VCS/CVS for a number of years -- but when a new software courted him with the stronger message that "your goals are my goals," he was able to overcome his indecision. Even then, and in spite of McVoy's claims on behalf of BK, he was still not ready to accept immediately. He was interested, yes, committed, no. Was this because BK was not GPLed, or is there more to the story?

BK can lock Torvalds into place with arguments that express his need to control Kernel development, review patches, and maintain architecture. His influence should not be compromised. BK emerges from the old problematizations, but with a new claim (interessement) to distributed yet centralized control. Translation is always a process before it is a result, so the early experience of Cort Dougan and his fellow developers at FSM labs helps nudge the indecision.

Callon describes interessement as an entity's actions "to impose and stabilize the identity of other actors it defines through its problematization." This is achieved by BK in respect of Torvalds, and for the Kernel code aligning with its desire to grow. BK also offers developers a full tree to work on, it will parade patches and merge them efficiently; plus, it has smart algorithms and a changeset feature to allow control. Its claim is to be a benevolent dictator's benevolent personal assistant. As shown in Figure 2, with the coming of BK, an OPP can be established – Torvalds will accept a VCS – and others too must pass through it. But what is missing from Figure 2, indeed from most of the debate, is

any question of the Constitution. As we will see, this will become a major challenge to the translation being attempted, and may in the end cut off an important set of actors.

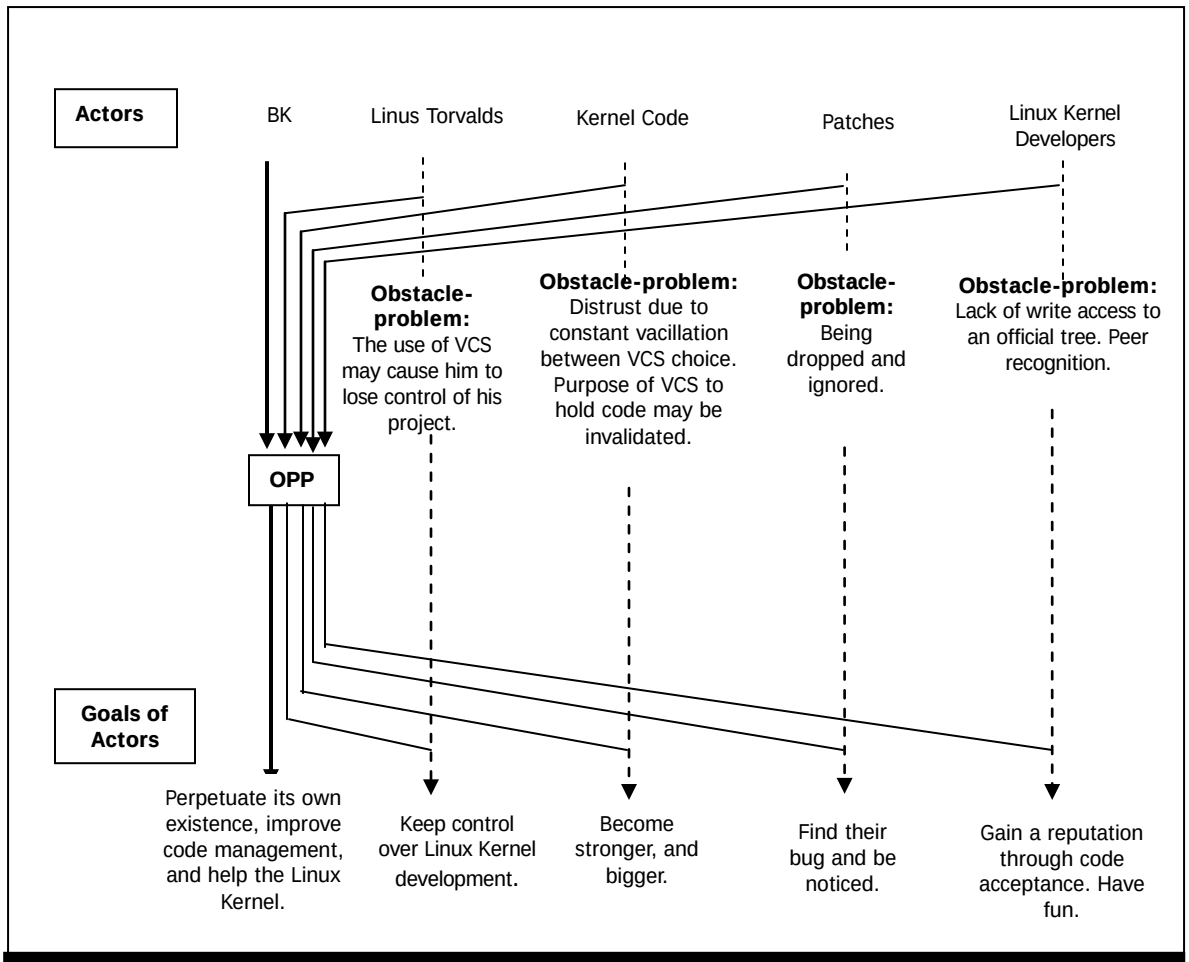

Figure 2. Actors, their Goals and Obstacles to Becoming Enrolled

5.3 Gathering allies – enrollment

For now, a new network is emerging with BK at the center, an OPP, but the putative allies must be enrolled so that they follow through with the plan. The answer to the question, "Can Torvalds be persuaded to adopt VCS as BK?" now has to be transformed into more certain statements of what will happen and what the actors will do – the machination.

Enrollment is seldom easy or direct. Any goal statement (cunning plan) that an actor wants to pursue must have a "margin of negotiation" (Latour, 1987, p. 208) so that other actors can interpret it, and adapt it to the local circumstances or to their desires. A margin of negotiation ensures that a statement has wider appeal, but carries with it the risk that it may be too changeable and control is lost. Torvalds must be enrolled, as must patches, developers, Kernel code, etc. These negotiations didn't take months, they took years. And as time passes, actors (and their networks) change.

The idea of VCS, what was once CVS and rejected by Torvalds, became BK – with a new network claiming technical excellence, efficiency, and dictator-friendly control. But now with an alien element embedded – no GPL – and thus no clear, unequivocal adherence to the Constitution. Torvalds' prior translation – his status as the OPP – shifted with his final acceptance, his enrollment in the network of BK, and BK's new position at the center of the Linux collective. This was offensive to many, and

neither Torvalds nor BK were prepared for the backlash. They had not expected constitutional niceties to become such a tricky margin of negotiation (Torvalds is not the first dictator with this problem). So interessement and its devices does not necessarily lead to enrollment. Indeed, in this case, enrollment was partial and harsh, with negotiations and concessions being made, as in the establishment of the Gateway or the rewording of the BK license. In all this, stronger and stronger references were made to the Constitution and to the ability of patches and their metadata to retain their fidelity to it. The Kernel code's status, too, was in question. "*Linux ceases to be free software when you require non-free software to contribute to it,*" said Alan Cox – there's not much of a margin for negotiation there.

5.4 The voices of VCS – mobilization

Mobilization means making mobile, and refers to bringing together (moving) actors that were previously dispersed, and aligning them to the new OPP. This is done through representatives, spokespersons for the parties involved. Some actors mostly speak for themselves, e.g., Torvalds, but some need spokespersons: McVoy for BK, some developers for CVS, Torvalds for the kernel, developers perhaps for the patches. The question to be asked is always, "Are the spokesmen (sic) representative?" (Latour, 2004). Actors may present themselves to represent others and to speak for them, but should we believe in their representativeness? As actors negotiate with allies through such representative spokespersons, it is important that they speak truthfully for the network they claim.

Specifically, in mobilization, the aim is to cut off competing translations and to reassemble actors in the way most beneficial for the OPP. But here, a number of such competing translations are heard from doubting developers speaking for code and patches. Appeals were made once again to the Constitution, which seemed to offer some guarantee of representation, but this is an unreformed type of politics that engages dictators – benevolent, albeit.

The collective made a choice about whether it wanted VCS or not. Two broad camps emerged, both in favor of VCS and acknowledging its legitimacy, but one favored the GPL'ed CVS (because it is constitutionally valid, or because it is technically suitable?) and the other, the more "efficient" (but commercially crippled) proprietary BK. This led to problems, but enough representatives emerged and gave voice for BK for it to become a core part of the collective. The Kernel and the patches agreed, through their spokespersons, and join the alliance; patches were paraded in ever increasing numbers, bugs were matched, Kernels went forth.

Torvalds is the interesting actor here. The coming of VCS into the collective, manifest as BK, was dependent on his approval, his translation. After much resistance, he not only accepted VCS use, but created an official BK Kernel tree and became a very public spokesman for BK, (and indirectly for the idea of VCS, if it can be separated from that of CVS). He was visibly mobilized, openly advocating BK on the LKML and participated in presentations and conferences where he justified his use of BK. Even in 2007, in the post-BK era, he is still justifying the pragmatic and technical case for BK while shrugging his shoulders at the "politics."[40]

But is the story brought to an end in this way – just "Torvalds has spoken"? We might rather say that it is BK that now claimed to speak for the other actors. To do so, it engaged them (problematization and interessement), seduced them (enrollment) and mobilized them (enough of them), achieving this through persuasion and example. Gaining agreement around a single goal (to serve Torvalds) allowed it to speak for everyone – in the name of Torvalds, the patches, and the unruly developers. It spoke, too, and with authority (though some circumspection – remember the hidden metadata) for Linux, the operating system kernel whose secrets it now held.

Our story, thus, ends with a triumphant, ascendant, sitting-next-to-Torvalds BK that has coerced, enticed, and maneuvered the recalcitrant actors and become a machination, "*transforming the juxtaposed set of allies into a whole that acts as one"* (Latour, 1987, pp. 128-129).

[40] http://www.youtube.com/watch?v=4XpnKHJAok8

5.5 Controversies and Breakdowns

But what happens if translations fail, if "translation becomes treason"? The collective translated in this version of the network may not be so durable or stable. The spokespersons negotiated the mobilization, but this may break down if the representatives do not continue to enjoy the following they claimed (their networks). If a representative is rejected, then we have what Callon calls a new controversy. Controversies lead to calls on the Constitution and issues of its infringement. The case provides a number of examples of controversy framed in such a way. We briefly address two such instances.

BK→CVS gateway – The gathering of forces – the machination – was not durable enough to keep some actors from dissenting. We see this unfold as the gateway created by BK to give CVS users access to Kernel code held by BK is rejected. Developers no longer use it, patches do not use it, and the gateway in this form became defunct, aligned with no network. Who should be blamed? Was this loss of representation caused by BK withholding metadata from CVS users, or by patches deciding that they did not see the appeal of this new network? The claim was that only a fragment (10 percent) of the metadata was missing, but this bred sufficient distrust among the developers, so the BK→CVS gateway failed to itself become an OPP.

BitKeeper License – The BK license was intended as an interessement device, a way to maintain the developers working with it and to reassure them that it was doing its best for them. However, the license, and in particular its change, created more dissidents than it did followers, even among some loyal Kernel developers, with strong appeals made to the Constitution. The translation, indeed, became treason by offering un-open source friendly restrictions.

6. Discussion

Linux development and general OS arrangements are effective in signalling that OS is something new and different, a new social invention or experiment, and offers many anthropological, organizational, and economic themes for its study. But, a more reflective analysis, for example within the perspectives of management science and organization theory, proves very confusing, as noted by Weber and others (Agerfalk and Fitzgerald, 2008; Fitzgerald, 2006; Weber, 2004) and badly needs a more sophisticated account of the open source model. The cathedral and the bazaar will not do as fundamental models.

What we do know from this case study is that open source delivers complex products and services, and along the way, it discovers new things out of controversies and through complex discussions. It is, as we have argued here, a hierarchy, but also a working *laboratory undertaken by a socio-technical collective* -- the site of social, organizational, and technical experiments to resolve complex matters of concern. Rather than being ruled by the simple norm of reciprocity (Gouldner, 1960), as any well functioning market, or the property rights of the firm, open source, in order to achieve and sustain over time its innovative experimentation and production, is composed by a set of *constituencies* of experts and specialists, but also technical artifacts and concepts, who commit to this because they share common values (under what we call the Constitution) and can align to common goals. Neither the anonymity and social distance that may characterize interactions over the net, nor the types of intricate and emotionally charged conversations and relations going on within the collective, have much in common with the often faceless and ephemeral encounters in a marketplace.

Our study of VCS uses public records available in the form of discussion threads, and an analysis within STS to highlight the role of technology (more generally of nonhumans, artifacts) in the laboratory's work of discovery and innovation. LKML threads allow us to see the collective at work as a minimal hierarchical organization (and struggling to retain that minimalness), while developing its capacity as a laboratory where experiments can be set up that may be able to transform a "matter of concern" into a "matter of fact" embodied in a robust machination, all undertaken under the Constitution.

We have shown that Linux and its (few) managers and architects do indeed share some characteristics of a formal organization and can learn from formal theories when facing classic problems of how to carry out efficient coordination of multiple activities using formal organizational mechanisms, like use of hierarchy, delegation, and vertical and horizontal information systems. The LKML threads tell us that Torvalds faced almost exactly the same problems a corporate manager would face, for example, when she needs to choose between centralization or decentralization of decision making as part of organizational redesign choices with technology (Applegate *et al*., 1988). Such models are able to predict what the available choices are, even their pros and cons, and the Internet and the myriad informational tools and uses it supports has not changed the nature of the challenge of formal organizational design in this respect, it has just extended enormously the realm of its application.

This ANT analysis of the restructuring of the collective shows, always through the publically available threads, not only the organizational and technical choices faced, but also the politics (politics *of* technology and politics *by* technology) deep inside the Linux laboratory-collective. We have shown that ANT can offer a language and model to carry out stakeholder analysis that highlights how VCS is certainly not "just a tool" and that the "let the code decide" credo of Linux means something deep, something that our approach has been able to reveal and articulate. Not only does BK maneuver (for a time) to the heart of the collective, making the top of the Linux organization a socio-technical hybrid -- Torvalds cannot exercise authority *de facto* without the new "close lieutenant" BK -- but also, VCS can be seen showing its own collective agency. Thinking of it as just a vertical *system* might be risky for the people who have to deal with it.

We argue that this ANT analysis and the laboratory-collective model, is distinctive because it takes the idea of organizing via and alongside technology well beyond an information processing understanding built on essentially functional accounts of technologies. Rather, it provides a micro analysis that lays bare the various political moves necessary to strengthen and sustain control and brings the participation of technology in the politics of the collective into focus. As our case indicates, managers, whether in open source projects or working in more conventional organizational structures, need technology as an ally. The story of VCS shows how negotiations with this potential ally occur, for example in the LKML, and as the framing of the technical is undertaken to place it within a strong machination.

But these threads tell us more. To be sure, painstaking discussions aimed at building consensus around the acquisition of a new technology (or responding to a new actor making claims to a network in our language) can be found in any business firm; but the crux of the controversies would be price, technical superiority, and, in many cases, power (Pfeffer, 1987). Here, the threads and their occasional flaming are also about the constitutionality of the choice of VCS. These conversations and controversies are what keeps together the background formative context (Ciborra and Lanzara, 1994) within which some hierarchy can be (to a degree and within some limits) made to work, smart tools can be accepted, patches organized, and bugs acknowledged. All of these activities occur in the peculiar way of open source when compared to the business firm, and are located in the complex formative contexts of quasi markets, public property, and individual incentives. We suggest that economic analyses of open source (Kollock, 1999; Lerner and Tirole, 2000; Lerner and Tirole, 2004; Meyer and Montagne, 2007; Sauer, 2007) that take for granted the latter formative context have missed a profound difference and have so far failed to deliver a convincing explanation of the inner workings of OS. The laboratory-collective model proposed here, with its explicit socio-technical and performative perspective, can help us see a little further, and, more particularly, deeper, into open source activity and help to reveal the fundamental forces that have driven such spectacular, and to some improbable, innovations.

7. Conclusion

In conclusion, we want to turn to two aspects that have emerged in our study and that might be taken into account in further research and practice in this domain.

The first is reflective and deals with the roles of threads and of researchers like us and many others exploring open source. Threads are vital in giving access to outsiders, again in striking contrast to the privacy of traditional business firm records. Threads represent the transparent memory of the collective, and allow unprecedented learning and reflection, helping outsiders to tap in, but also providing for feedback, analyses, and reflections. A thread is an actor in itself, a device of translation – interessement and enrolment – one to which we, the authors, have succumbed. They bring to the collective an essential capacity for reflexivity and learning, noting that the messages that flow through the LKML don't just relate to the code proper, but are also patches for the collective itself – at times even potential patches for the Constitution (amendments, we might say?). Wisely, as in most constitutional polities, it is hard to change the constitution and requires strong degrees of agreement.

This links to the second more normative aspect we see emerging from this work. This analysis of "what is going on," seen from the vantage point of a performative ontology addressing a process of *innovation* in the face of change, *machination* to hold together multiple interests, and the *Constitution* to retain allegiance and allow accommodations to be negotiated among the multiple parties, suggests, indeed, that governance for the collective is not in the hands of humans alone. The unprecedented intimacy of people and code, of machines and controversies, of translation and treason in the collective, provides the ideal setting – the ideal *Laboratory* -- to carry out an experiment in constitutional reform: a case study, perhaps, of Latour's idea of the "*Parliament of Things*" (Latour, 1993; Latour, 2004) and Callon et al.'s (2009) "*Technical Democracy*."

Finally, our analysis of the Linux organization under stress, both when devising a new structure supported by new information systems and when struggling to find constitutionally correct and enduring ways to implement it, shows that any open source-like arrangement must solve simultaneously the problems of efficient formal organization design, be prepared to run experiments in a Laboratory where matters of concern can be opened up, and function as a parliament of things. This leads us to our future research question: Can these three distinct elements be traced and analyzed in other emerging open source-like arrangements, for example, in genomics (see www.ensembl.org), or drug discovery (see www.osdd.net) (Munos, 2006), where the institutional formative context, the technology, and the rules of the collective may be significantly different?

References

Agerfalk, P. and B. Fitzgerald (2008) "Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy", *MIS Quarterly,* 32 (2), pp. 385-400.

Akrich, M. (1992) "The De-Scription of Technical Objects" in *Shaping Technology/Building Society*, (Bijker, W. E. and J. Law eds) MIT Press, Cambridge, MA.

Andrews, J. and L. McVoy (2002) *Interview: Larry Mcvoy* KernelTrap.org Address: http://Kerneltrap.org/node.php?id=222.

Anvin, H. P. (2004 - Thu 22nd July) *Re: Linux-Kernel CVS Gateway?* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0407.2/0490.html.

Applegate, L. M., J. I. Cash and D. Q. Mills (1988) "Information Technology and Tomorrow's Manager", *Harvard Business Review,* Nov-Dec pp. 128-136.

Bach, P. (2009). Design Information sharing across multiple knowledge systems in a FLOSS community. iConference, University of North Carolina-Chapel Hill, USA.

Barr, J. and L. McVoy (2003) *Larry Mcvoy on Bitkeeper, Kernel Development, Linus Torvalds & Bruce Perens* LinuxWorld Address: http://www.linuxworld.com/story/32618_4.htm.

Benkler, Y. (2004) "Sharing Nicely: On Shareable Goods and the Emergence of Sharing as a Modality of Economic Production", *Yale Law Journal,* 114 (273-358).

Berglund, M. (2000 - Mon 11th Sep) *[Announce] Darkstar Development Project* University of Indiana Address: http://www.uwsg.iu.edu/hypermail/linux/Kernel/0009.1/0472.html.

Berliner, B. (1990) "CVS II: Parallelizing Software Development". in *Proceedings of the USENIX Winter 1990 Technical Conference,* Washington D.C.

Bolinger, D. and T. Bronson (1995) *Applying RCS and SCCS: From Source Control to Project Control,* O'Reilly & Associates,

Bonaccorsi, A. and C. Rossi (2003) "Licensing Schemes in the Production and Distribution of Open Source Software: An Empirical Investigation", *SSRN working paper available online at: http://ssrn.com/abstract=432641.*

Bradford, J. (2003 - Sun 2nd March) *Re: Bitbucket: GPL-ed Kitbeeper Clone,* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0411.html.

Bruns, A. (2008) *Blogs, Wikipedia, Second Life and Beyond: From Production to Produsage,* Peter Lang Pub Inc, New York.

Callon, M. (1986a) "The Sociology of an Actor-Network" in *Mapping the Dynamics of Science and Technology*, (Callon, M., J. Law and A. Rip eds) Macmillan, London.

Callon, M. (1986b) "Some Elements of a Sociology of Translation: Domestication of the Scallops and Fishermen of St. Brieuc Bay" in *Power, Action and Belief: A New Sociology of Knowledge?*, (Law, J. ed.) (32) Routledge & Kegan Paul, London, pp. 196-233.

Callon, M. (1999) "Actor-Network Theory - the Market Test" in *Actor Network Theory and After*, (Law, J. and J. Hassard eds) Blackwell Publishers / The Sociological Review, Oxford, pp. 181-195.

Callon, M., P. Lascoumes and Y. Barthe (2009) *Acting in an Uncertain World : An Essay on Technical Democracy* MIT Press, Cambridge, Mass. .

Callon, M. and J. Law (1995) "Agency and the Hybrid Collectif", *South Atlantic Quarterly,* 94 (2), pp. 481-507.

Callon, M., Y. Millo and F. Muniesa (2007) *Market Devices,* Blackwell, London.

Callon, M. and F. Muniesa (2005) "Economic Markets as Calculative Collective Devices", *Organization Studies,* 26 (8), pp. 1229-1250.

Chesney, T. (2006) "An Empirical Examination of Wikipedia's Credibility", *FirstMonday,* 11 (11).

Ciborra, C. and G. F. Lanzara (1994) "Formative Contexts and Information Technology", *Accounting, Management and Information Technology,* 4 pp. 611-626.

Ciborra, C. U. (1993) *Teams Markets and Systems,* Cambridge University Press, Cambridge.

Ciborra, C. U. (Ed.) (2000) *From Control to Drift,* Oxford University Press, Oxford.

Clegg, S., M. Kornberger and C. Rhodes (2005) "Learning/Becoming/Organizing", *Organization,* 12 (2), pp. 147-167.

Clemm, G. (1989) "Replacing Version-Control with Job-Control". in *Proceedings of the 2nd International Workshop on Software Configuration Management,* Princeton, New Jersey, United States, pp. 162-169.

Collins-Sussman, B., B. W. Fitzpatrick and C. M. Pilato (2002) *Version Control with Subversion,*

O'Reilly Media,
Collins, B. (2003 - Tue 11th March) *Re: [Announce] BK->CVS (Real Time Mirror)* University of Indiana Address: http://www.ussg.iu.edu/hypermail/linux/Kernel/0303.1/0894.html.
Cornford, T., C. Ciborra and M. Shaikh (2005) "Do Penguins Eat Scallops?", *European Journal of Information Systems,* 14 (5), pp. 518–521.
Cox, A. (2000 - Wed 13th Sep) *Re: Proposal: Linux Kernel Patch Management System* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0009.1/1076.html.
Crowston, K. and J. Howison (2006) "Hierarchy and Centralization in Free and Open Source Software Team Communications", *Knowledge, Technology, and Policy,* 18 (4), pp. 65-85.
Davis, G. B. and M. B. Olson (1985) *Management Information Systems: Conceptual Foundations, Structure and Development,* (2) McGraw-Hill, New York.
De Paoli, S. and V. D'Andrea (2008) "How Artifacts Rule Web-Based Communities: Practices of Free Software Development", *International Journal of Web Based Communities* 4(2), pp. 199-219.
Demil, B. and X. Lecocq (2006) "Neither Market nor Hierarchy nor Network: The Emergence of Bazaar Governance", *Organization Studies,* 27 (10), pp. 1447-1466.
Fitzgerald, B. (2006) "The Transformation of Open Source Software", *MIS Quarterly,* 30 (3), pp. 587-598.
Fitzgerald, B. and G. Bassett (2003) "Legal Issues Relating to Free and Open Source Software" in *Legal Issues Relating to Free and Open Source Software: Essays in Technology Policy and Law Volume 1*, (Fitzgerald, B. and G. Bassett eds) Queensland University of Technology: School of Law, pp. 11-36.
Fogel, K. (1999) *Open Source Development with CVS,* Coriolis Open Press, Scottsdale, AZ.
Forge, S. (2000) "Open Source: The Economics of Giving Away Stuff, and Software as a Political Statement", *The Journal of Policy, Regulation and Strategy for Telecommunications Information and Media,* 2 (1), pp. 5-7.
Galbraith, J. R. (1974) "Organization Design: An Information Processing View", *Interfaces,* 4 (3), pp. 28-36.
Galbraith, J. R. (1977) *Organisation Design,* Addison-Wesley,
Gall, T. (2002- Fri 4th Oct) *New BK License Problem?* University of Indiana Address: http://www.uwsg.iu.edu/hypermail/linux/Kernel/0210.0/1496.html.
Garzik, J. (2003 - Sat 1st March) *Re: Bitbucket: GPL-ed Kitbeeper Clone* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0147.html.
Gouldner, A. W. (1960) "The Norm of Reciprocity: A Preliminary Statement", *American Sociological Review,* 25 pp. 161-178.
Hanseth, O. and E. Monteiro (1997) "Inscribing Behaviour in Information Infrastructure Standards", *Accounting, Management and Information Technology,* 7 (4), pp. 183-211.
Hars, A. and S. Ou (2002) "Working for Free? Motivations for Participating in Open-Source Projects", *International Journal of Electronic Commerce,* 6 (3), pp. 25–39.
Hellwig, C. (2003 - Sat 1st March) *Re: Bitbucket: GPL-ed Kitbeeper Clone* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0092.html.
Henkel, J. (2004) *Patterns of Free Revealing - Balancing Code Sharing and Protection in Commercial Open Source Development* MIT Working Paper Address: http://opensource.mit.edu/papers/henkel2.pdf.
Henson, V. and J. Garzik (2002) "Bitkeeper for Kernel Developers". in *Ottawa Linux Symposium,* Ottawa, Ontario Canada, June 26th–29th, 2002, pp. 197-212,
Jensen, C. and W. Scacchi (2005). Collaboration, Leadership, Control, and Conflict Negotiation Processes in the NetBeans.org Open Source Software Development Community. Hawaii International Conference of Systems Science, Waikola Village, Hawaii, USA.
Joerges, B. and B. Czarniawska (1998) "The Question of Technology, or How Organizations Inscribe the World", *Organization Studies,* 19 (3), pp. 363-385.
Keen, P. (1981) "Information Systems and Organizational Change", *Communications of the ACM,* 24 (1), pp. 24-33.
Kilpi, T. (1997) "New Challenges for Version Control and Configuration Management: A Framework and Evaluation", *IEEE Computer,* (1st Euromicro Working Conference on Software Maintenance and Reengineering (CSMR '97)), pp. 33-41.
Koch, S. and G. Schneider (2002) "Effort, Cooperation and Coordination in an Open Source Software

Project: Gnome", *Information Systems Journal,* 12 (1), pp. 27-42.
Kollock, P. (1999) "The Economics of Online Cooperation: Gift and Public Goods in Cyberspace" in *Communities in Cyberspace*, (Smith, Marc and Kollock eds) Routledge, London, pp. 220-246.
Lakhani, K. R. and R. G. Wolf (2005) "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects" in *Perspectives on Free and Open Source Software*, (Feller, J., B. Fitzgerald, S. Hissam and K. R. Lakhani eds) MIT Press.
Lanzara, G. F. and M. Morner (2005) "Artifacts Rule! How Organizing Happens in Open Source Projects" in *Actor-Network Theory and Organizing*, (Czarniawska, B. and T. Hernes eds) Copenhagen Business School Press, Copenhagen, Denmark.
Latour, B. (1986) "The Powers of Association" in *Power, Action and Belief. A New Sociology of Knowledge? Sociological Review Monograph 32*, (Law, J. ed.) Routledge & Kegan and Paul, London, pp. 264-280.
Latour, B. (1987) *Science in Action: How to Follow Scientists and Engineers through Society,* Bantam, New York.
Latour, B. (1988) "The Prince for Machines as Well as for Machinations" in *Technology and Social Process*, (Elliot, B. ed.) Edinburgh University Press, Edinburgh, pp. 20-43.
Latour, B. (1991) "Technology Is Society Made Durable" in *A Sociology of Monsters: Essays on Power, Technology and Domination*, (Law, J. ed.) Routledge, London.
Latour, B. (1993) *We Have Never Been Modern,* Harvester Wheatsheaf, Hemel Hempstead.
Latour, B. (1996) *Aramis or the Love of Technology,* Harvard University Press,
Latour, B. (1999a) *Pandora's Hope,* Harvard University Press, Cambridge, Massachusetts.
Latour, B. (1999b) *Pandora's Hope: Essays on the Reality of Science Studies,* Harvard University Press, Cambridge, MA.
Latour, B. (2004) *Politics of Nature : How to Bring the Sciences into Democracy,* Harvard University Press,
Latour, B. (2005) *Reassembling the Social: An Introduction to Actor-Network-Theory (Clarendon Lectures in Management Studies),* Oxford University Press, Oxford.
Law, J. (1991) *Sociology of Monsters: Essays on Power, Technology and Domination* Routledge, London.
Law, J. (1992) "Notes on the Theory of the Actor-Network: Ordering, Strategy and Heterogeneity", *Systems Practice,* 5 (4), pp. 379-393.
Law, J. (1997) *Traduction/Trahison - Notes on ANT,* Department of Sociology Lancaster University Address: http://www.comp.lancs.ac.uk/sociology/stslaw2.html.
Law, J. (1999) "After ANT: Topology, Naming and Complexity" in *Actor Network Theory and After*, (Law, J. and J. Hassard eds) Blackwell and the Sociological Review, Oxford and Keele.
Law, J. and M. Callon (1988) "Engineering and Sociology in a Military Aircraft Project: A Network Analysis of Technological Change", *Social Problems,* 35 (3), pp. 284-297.
Law, J. and A. Mol (1995) "Notes on Materiality and Sociality", *The Sociological Review,* 43 pp. 274-294.
Lerner, J. and J. Tirole (2000) "The Simple Economics of Open Source Code". in pp. 1-40, National Bureau of Economic Research Working Paper Series.
Lerner, J. and J. Tirole (2002) "Some Simple Economics of the Open Source", *The Journal of Industrial Economics,* 2 pp. 197-234.
Lerner, J. and J. Tirole (2004) "The Economics of Technology Sharing: Open Source and Beyond," Negotiation, Organizations and Markets Unit, Research Paper Series 04-35, *Harvard Business School.*
Lessig, L. (1999) *Code and Other Laws of Cyberspace,* Basic Books.
Ljungberg, J. (2000) "Open Source Movements as a Model for Organizing", *European Journal of Information Systems,* 9 (4), pp. 208-216.
Lopez-Fernandez, L., G. Robles and J. Gonzalez-Barahona (2004) "Applying Social Network Analysis to the Information in CVS Repositories". in *International Workshop on Mining Software Repositories (MSR),* Edinburgh, Scotland, pp. 101-105, IEEE.
Luhmann, N. (1984) *Soziale Systeme,* Suhrkamp, Frankfurt.
Machek, P. (2003 - Mon 3rd March) *Re: Bitbucket: GPL-ed Kitbeeper Clone* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0422.html.
Machek, P. (2003 - Wed 26th Feb) *Bitbucket: GPL-ed Bitkeeper Clone* University of Indiana

Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0302.3/0931.html.
Markus, M. L. (1983) "Power, Politics and MIS Implementation", *Communications of the ACM,* 26 (6), pp. 430-444.
Markus, M. L., B. Manville and C. E. Agres (2000) "What Makes a Virtual Organization Work?", *Sloan Management Review,* 42 (1), pp. 13-26.
Markus, M. L. and D. Robey (1988) "Information Technology and Organizational Change", *Management Science,* 34 (5), pp. 583-598.
McVoy, L. (1999 - Sun 21st Feb) *Revision Control for the Kernel (Bitkeeper)* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/9902.2/0812.html.
McVoy, L. (2002 - Fri 4th Oct) *Re: New BK License Problem?* University of Indiana Address: http://www.uwsg.iu.edu/hypermail/linux/Kernel/0210.0/1500.html.
McVoy, L. (2003 - Mon 17th March) *BK->CVS Is Live* University of Indiana Address: http://www.ussg.iu.edu/hypermail/linux/Kernel/0303.2/0219.html.
McVoy, L. (2003 - Sat 1st March) *Re: Bitbucket: GPL-ed Bitkeeper Clone* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0052.html.
McVoy, L. (2003 - Sat Mar 01) *Re: Bitbucket: GPL-ed Bitkeeper Clone* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0057.html.
McVoy, L. (2003 - Sun 16th March) *Re: [Announce] BK->CVS (Real Time Mirror)* University of Indiana Address: http://www.ussg.iu.edu/hypermail/linux/Kernel/0303.2/0121.html.
McVoy, L. (2003 - Tue 11th March) *Re: [Announce] BK->CVS (Real Time Mirror)* University of Indiana Address: http://www.ussg.iu.edu/hypermail/linux/Kernel/0303.1/0900.html.
McVoy, L. (2003 - Wed 12th March) *Re: [Announce] BK->CVS (Real Time Mirror)* University of Indiana Address: http://www.ussg.iu.edu/hypermail/linux/Kernel/0303.1/1242.html.
McVoy, L. (2003 - Wed Mar 12) *Re: [Announce] BK->CVS (Real Time Mirror)* University of Indiana Address: http://www.ussg.iu.edu/hypermail/linux/Kernel/0303.1/1134.html.
Meyer, M. and F. Montagne (2007) "Open Source Software and the Self-Governed Community", *Revue D Economie Politique,* 117 (3), pp. 387-405.
Miller, D. S. (1997 - Sun, 14th Sep) *First Vger CVS 2.1.X Kernel Source Repository Snapshot* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/9709.1/0432.html.
Moglen, E. (1999) "Anarchism Triumphant: Free Software and the Death of Copyright", *First Monday,* 8 (4).
Molnar, I. (2002 - Sun 6th Oct) *BK Metadata License Problem?* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0210.0/1918.html.
Moody, G. (2001) *Rebel Code: Linux and the Open Source Revolution,* Penguin, London.
Munos, B. (2006) "Outlook: Can Open-Source R&D Reinvigorate Drug Research?", *Nature Reviews Drug Discovery,* 5 pp. 723-729.
O'Mahony, C. S. (2003) "Guarding the Commons: How Community Managed Software Projects Protect Their Work", *Research Policy,* 32 pp. 1168-1198.
O'Mahony, S. (2005). The Role of a Boundary Institution in Reconciling Convergent and Divergent Logics. American Sociological Association, Philadelphia, PA, USA.
O'Mahony, S. and B. Bechky (2008). "Boundary Organizations: Enabling Collaboration among Unexpected Allies." *Administrative Science Quarterly* 53: 422–459.
Orlikowski, W. and S. Iacono (2001) "Desperately Seeking The "IT" In IT Research - a Call to Theorizing the IT Artifact", *Information Systems Research,,* 12 (2), pp. 121-134.
Ouchi, W. (1980) "Markets, Bureaucracies and Clans", *Administrative Science Quarterly,* 25 pp. 120-142.
Paoli, S. D., M. Teli, et al. (2008). "Free and open source licenses in community life: Two empirical cases." *First Monday* 13(10).
Perens, B. (1999) "The Open Source Definition" in *Open Sources: Voices from the Open Source Revolution*, (DiBona, C., S. Ockman and M. Stone eds) O'Reilly, CA.
Pfeffer, J. (1987) "A Resource Dependence Perspective on Intercorporate Relations" in *Intercorporate Relations: The Structural Analysis of Business*, (Mizruchi, M. and M. Schwartz eds) Cambridge University Press, Cambridge.
Pollock, N. and J. Cornford (2004) "Customising Industry Standard Computer Systems for Universities: ERP Systems and the University as a “Unique” Organisation", *Information*

Technology and People, 17 (1), pp. 31-52.
Raymond, E. (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary,* O'Reilly & Associates, Sebastopol, California.
Rochkind, M. J. (1975) "The Source Code Control System", *IEEE Transaction Software Engineering,* 1 (4), pp. 364-370.
Rose, J. and M. Jones (2004) "The Double Dance of Agency: A Socio-Theoretic Account of How Machines and Humans Interact". in *ALOIS 2004 - Action in Language, Organisations and Information Systems,* Linköping - Sweden,
Sauer, R. M. (2007) "Why Develop Open-Source Software? The Role of Non-Pecuniary Benefits, Monetary Rewards, and Open-Source Licence Type", *Oxford Review of Economic Policy* 23 (4), pp. 605-619.
Suber, P. (2006) "Creating an Intellectual Commons through Open Access" in *Understanding Knowledge as a Commons: From Theory to Practice*, (Hess, C. and E. Ostrom eds) MIT Press.
Suber, P. (2007) "Trends Favoring Open Access, Cyberinfrastructure Technology Watch (Ctwatch)", *Scholarly Communications & Cyberinfrastructure,* 3 (3).
Tichy, W. F. (1985) "RCS - a System for Version Control.", *Software - Practice and Experience,* 15 (7), pp. 637-654.
Torvalds, L. (2002 - Sat 20th April) *Re: [Patch] Remove Bitkeeper Documentation from Linux Tree* University of Indiana
Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0204.2/1018.html.
Torvalds, L. (2002 - Tue 5th Feb) *Linux-2.5.4-Pre1 - Bitkeeper Testing* University of Indiana
Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0202.0/0989.html.
Torvalds, L. (2003) *Managing Kernel Development* Linux Lunacy Geek Cruise
Address: http://www.linuxjournal.com//article.php?sid=7272.
Torvalds, L. (2003 - Fri 7th March) *Re: Bitbucket: GPL-ed Kitbeeper Clone* University of Indiana
Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/1932.html.
Torvalds, L. (2003 - Tue 4th March) *Re: Bitbucket: GPL-ed *Notrademarkhere* Clone* University of Indiana Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/0303.0/0695.html.
Torvalds, L. and C. Schlenter (1995, 1996) *Re: CVS, Linus, and Us* University of Indiana
Address: http://www.uwsg.indiana.edu/hypermail/linux/Kernel/9602/0800.html.
Tuomi, I. (2001) "Internet, Innovation, and Open Source: Actors in the Network", *First Monday,* 6 (1).
von Hippel, E. and G. von Krogh (2003) "Open Source Software and the “Private-Collective” Innovation Model: Issues for Organization Science", *Organization Science,* 14 (2), pp. 209-223.
Weber, S. (2004) *The Success of Open Source,* Harvard University Press,
Weber, S. (2005) "The Political Economy of Open Source and Why It Matters" in *Digital Formations: IT and New Architectures in the Global Realm*, (Latham, R. and S. Sassen eds) Princeton University Press, Princeton, New Jersey, pp. 178-212.
Weick, K. E. (1979) *The Social Psychology of Organizing,* (2nd) Addison-Wesley, Reading, MA.

About the Authors

Tony CORNFORD is Senior Lecturer in Department of Management at the London School of Economics and Political Science. His research interests are in health information systems particularly those supporting the use of drugs and medicines and open source software processes and business models. Recent research projects have included evaluation of a national programme for implementation of electronic patient records in England and a study of the electronic transmission of prescriptions.

Maha SHAIKH is a Research Associate at the London School of Economics and Political Science (LSE). She holds a PhD in Information Systems from the LSE. Her thesis focused on the dynamic of learning and organizing in open source collectives. She has been affiliated with the University of Limerick where her work revolved around a number of projects, including the OPAALS project, at Lero with Professor Brian Fitzgerald. At the LSE she was Research Officer to Professor Leslie Willcocks studying the relationship of open source to outsourcing, open innovation and open business models. She is currently a visiting lecturer at the Warwick Business School. Dr Shaikh has a forthcoming co-authored book on the secondary adoption of open source by the public sector published by the MIT Press. She has published and presented at all the main information systems conferences including ECIS, ICIS, AMCIS, HICSS, and her own special interest group, the OSS conference.

Claudio U. CIBORRA (1951-2005) was a Professor of Information Systems at the London School of Economics and Political Science. He was a leading researcher in the areas of information systems, management and strategy. His earlier work dealt with transaction cost and information technology (his *Teams, Markets and Systems* is in its third printing with Cambridge University Press). In the 1990's the focus of his research shifted to the issues of learning and the implementation of IT systems. He sat on the Editorial Board of the most important information systems journals in Europe, and also in journals in the domain of organization theory. He was a keynote speaker at various international information systems conferences, and worked with many corporations, public authorities, foundations, and intergovernmental bodies worldwide.